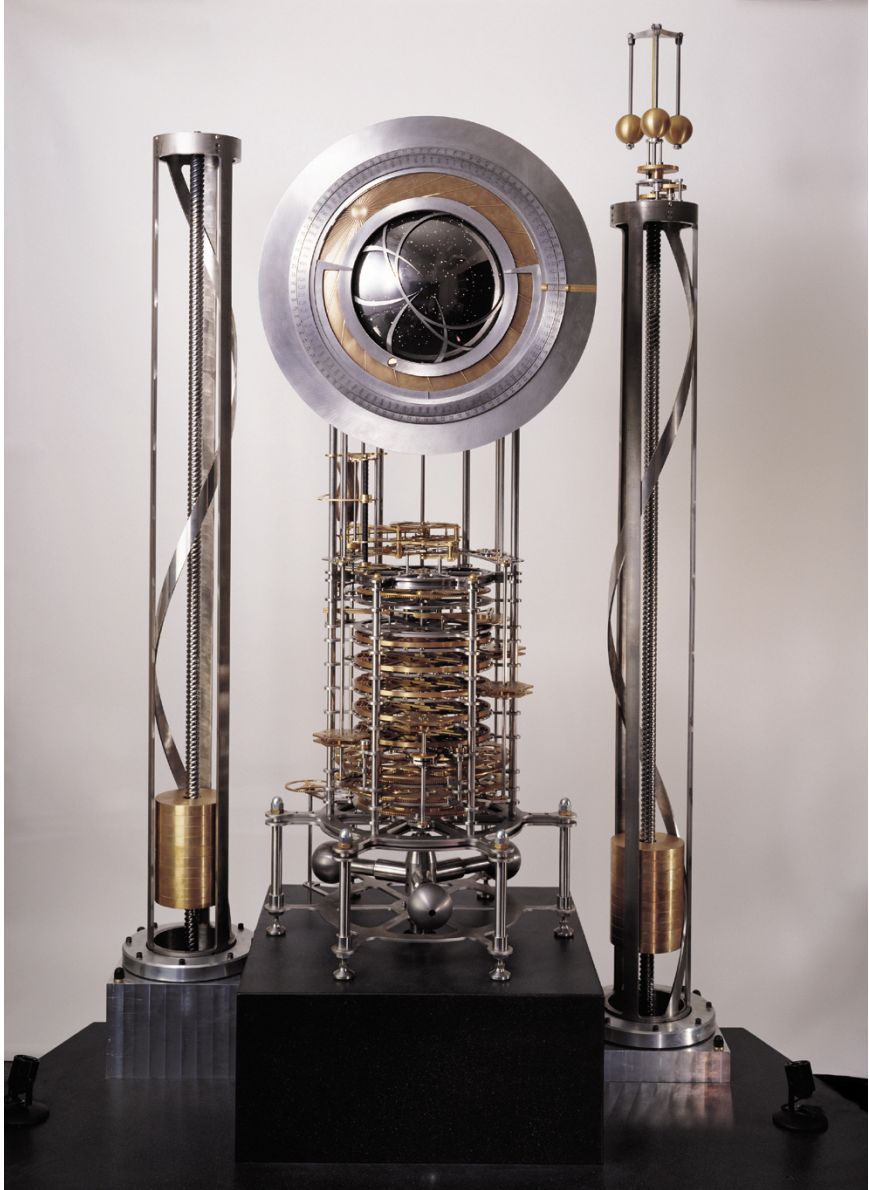
How do you even test that?

A software QA primer

Sean Dague
dague.net
@sdague



Disclaimers

- This topic is bigger than fits in this lecture
- All code shown will be python
- Techniques should work in multiple languages
- Have build/worked with versions of these approaches in:
  - bash
  - python
  - ruby
  - Perl
  - C
  - Java
  - C#

What is the purpose of testing?

Make Testing Great Again!

- Writing tests is extra work
- It has to be easy for people to write tests for new functions
- It has to be easy for people to run tests to feel worth writing tests
- Ideally most tests run on people's laptops
- Test failures should be repeatable

You don't have to go it alone

- Java - junit
- Python - unittest
- Perl - Test.pm
- C - autotest (part of autoconf / automake)
- Nearly every language environment has test framework to get running with

The simplest Test File: test_mystuff.py

```
import unittest

class TestMyTest(unittest.TestCase):

    def setUp(self):
        …

    def test_something(self):
        …
        self.assertEqual(a, b)

    def test_some_other_thing(self):
        …
        self.assertRaises(exception.Foo,
            somefunc, arg1, arg2)

    def tearDown(self):
        …
```

- Naming is magical, and important
  - Filename must start with test.*
  - Test classes must start with Test.*
  - Test functions must start with test.*
- run order: setUp → test_foo → tearDown for every test
- all tests run in a single process, global state changes will not automatically be reset
- run order of test_foo vs. test_foo2 is not defined, plan accordingly
- *all test systems will have caveats, read up on them*

Actually running tests

- What has to be installed?
- What are the dependencies?
- Testing different environments?

Is it even software?

```
def sum(*args):
    res = 0
    for arg in args:
       res += agr
    return res
```

- Compilers help in compiled languages
- In dynamic languages, you need linters
- for python: flake8 / pylint

Writing testable code

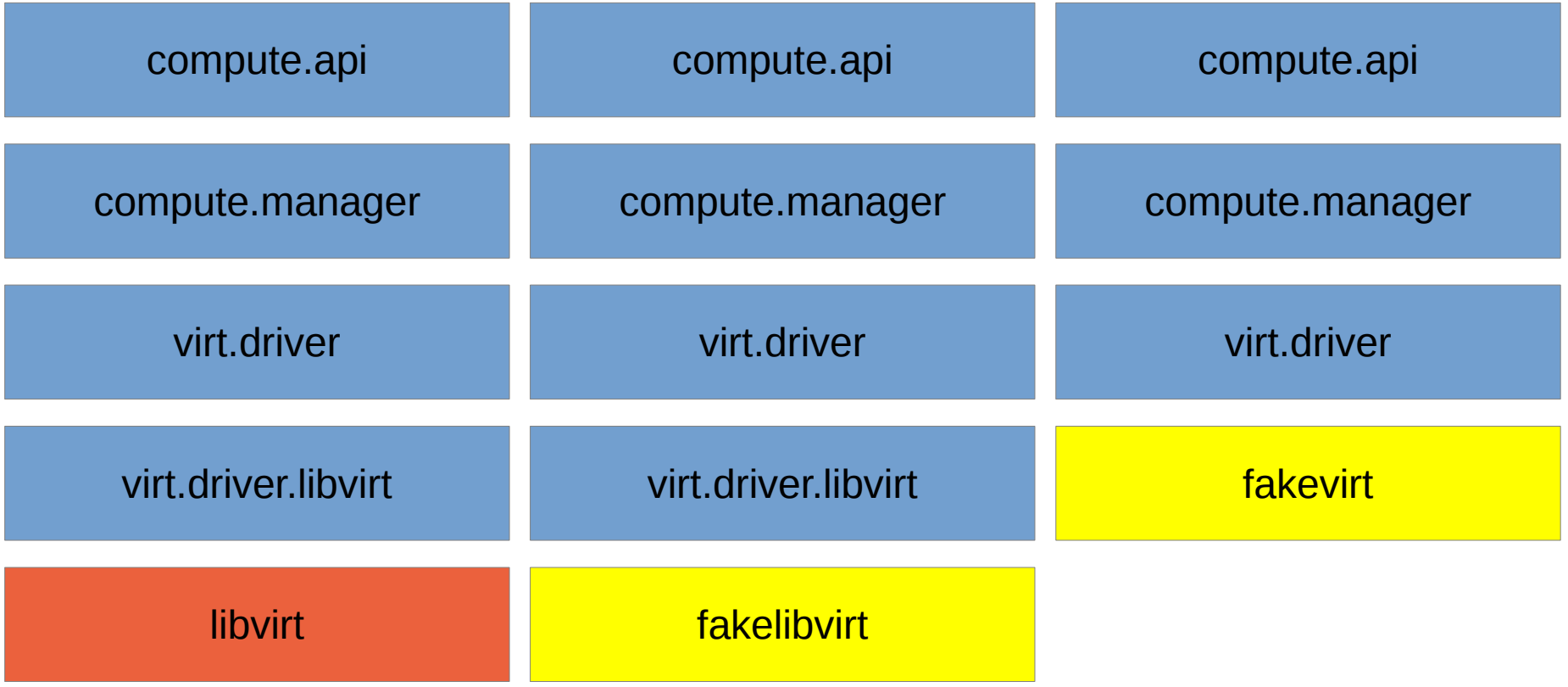
- What's perfectly testable code?
  - Limited inputs
  - Limited branches
  - No side effects - no state saving, no communication to other systems
  - AKA - LISP
- This is < 5% of all interesting software
- What do we do for the other 95%?

Fakes & Mocks

- Build a fake implementation of an abstraction layer
- Load and run for an entire test class
- Pros:
  - Easy to replace in testing
  - Can respond with anything you want, including odd errors that are hard to get in production
- Cons:
  - Only as accurate as you make it
  - Can easily become a complex simulator that has it's own bugs

- Dynamically replace function calls
- Typically done **per** test method
- Pros:
  - Very precise isolation
  - Don't need to build a whole fake
  - Really easy to do fault injection
- Cons:
  - Can very easily drift from real behavior
  - Can make refactoring hard

Fakes



Mocking in action

when called, return 11, always

```
@mock.patch.object(objects.Service, 'get_minimum_version',
                   return_value=11)
def test_validate_auto_or_none_network_request_old_computes(self, mock_get_ver):

    """Tests that the network request is nulled out when the minimum
       nova-compute is not running new enough code to support 'auto'.
    """
    req_nets = objects.NetworkRequestList(
        objects=[objects.NetworkRequest(network_id='auto')])

    self.assertIsNone(
        self.controller._validate_auto_or_none_network_request(req_nets))

    mock_get_ver.assert_called_once_with(mock.ANY, 'nova-compute')
```

Reference to mock

test that this was called exactly once with these parameters

Integration Testing

- Assemble the whole system into some workable whole
- Run some representative set of tests on it
- Ensure that all the individual working parts build a working whole

Bill Sempf
@sempf

Follow

QA Engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 999999999 beers. Orders a lizard. Orders -1 beers. Orders a sfdeljknesv.

RETWEETS	LIKES
28,900	19,217

1:56 PM - 23 Sep 2014

29K 19K

```
2016-10-30 16:52:19,208 [DEBUG] root: <class 'arwn.vendor.RFXtrx.SensorEvent'> device=[<class
'arwn.vendor.RFXtrx.RFXtrxDevice'> type='PCR800' id='96:00'] values=[('Battery numeric', 9), ('Rain Rate (mm/hr)',
0.99), ('Rain Total (mm)', 422.7), ('Rssi numeric', 6)]
2016-10-30 16:52:22,467 [DEBUG] root: <class 'arwn.vendor.RFXtrx.ControlEvent'> device=[<class
'arwn.vendor.RFXtrx.LightingDevice'> type='HomeEasy EU' id='2aa82aa:11'] values=[('Command', 'Set group level'), ('Dim
level', 68), ('Rssi numeric', 2)]
2016-10-30 16:52:22,468 [DEBUG] PidFile: <pid.PidFile object at 0x762bb1e8> closing pidfile:
/home/sdague/code/arwn/arwn.pid
2016-10-30 16:52:22,469 [ERROR] root: Something went wrong!
Traceback (most recent call last):
  File "/home/sdague/code/arwn/.venv/local/lib/python2.7/site-packages/arwn-1.0.0-py2.7.egg/arwn/cmd/collect.py", line
80, in main
    event_loop(config)
  File "/home/sdague/code/arwn/.venv/local/lib/python2.7/site-packages/arwn-1.0.0-py2.7.egg/arwn/cmd/collect.py", line
67, in event_loop
    dispatcher.loopforever()
  File "/home/sdague/code/arwn/.venv/local/lib/python2.7/site-packages/arwn-1.0.0-py2.7.egg/arwn/engine.py", line 186,
in loopforever
    packet.from_packet(event.device.pkt)
  File "/home/sdague/code/arwn/.venv/local/lib/python2.7/site-packages/arwn-1.0.0-py2.7.egg/arwn/engine.py", line 76, in
from_packet
    self.bat = packet.battery
AttributeError: 'Lighting2' object has no attribute 'battery'
2016-10-30 16:52:22,473 [DEBUG] PidFile: <pid.PidFile object at 0x762bb1e8> closing pidfile:
/home/sdague/code/arwn/arwn.pid
```

The journey of a thousand miles begins with a single step

phue.py

- Python interface to Hue light bulbs
- Constraints:
  - Single file
  - No external requirements
  - Has to work in python 2 & 3
- No existing testing at all

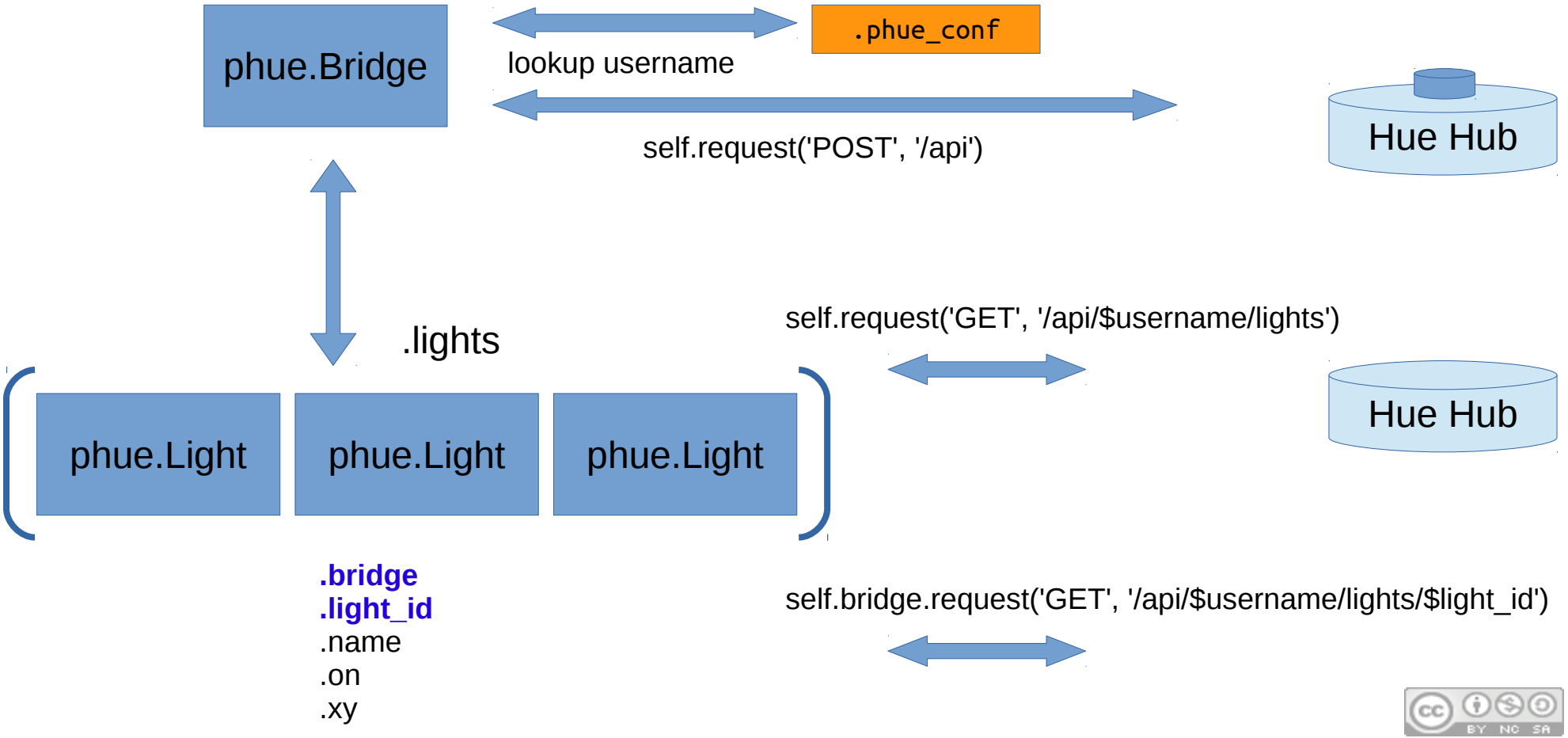
First test

```
import testtools

import phue  # noqa


class TestImport(testtools.TestCase):

    def test_import_works(self):
        pass
```

Test runner – tox.ini

```
[tox]
envlist = pep8, py27, py35
skip_missing_interpreters = True

[testenv]
setenv =
    LANG=en_US.UTF-8
    PYTHONPATH = {toxinidir}
commands =
     py.test -v --timeout=30 --duration=10 --cov=phue --cov-report html {posargs}
deps =
     -r{toxinidir}/test-requirements.txt

[testenv:pep8]
deps = flake8
basepython = python3
commands =
     flake8 phue.py

[flake8]
ignore = E501
exclude =  .venv,.git,.tox,dist,doc,*lib/python*,*egg,build
```

What's this software actually look like?

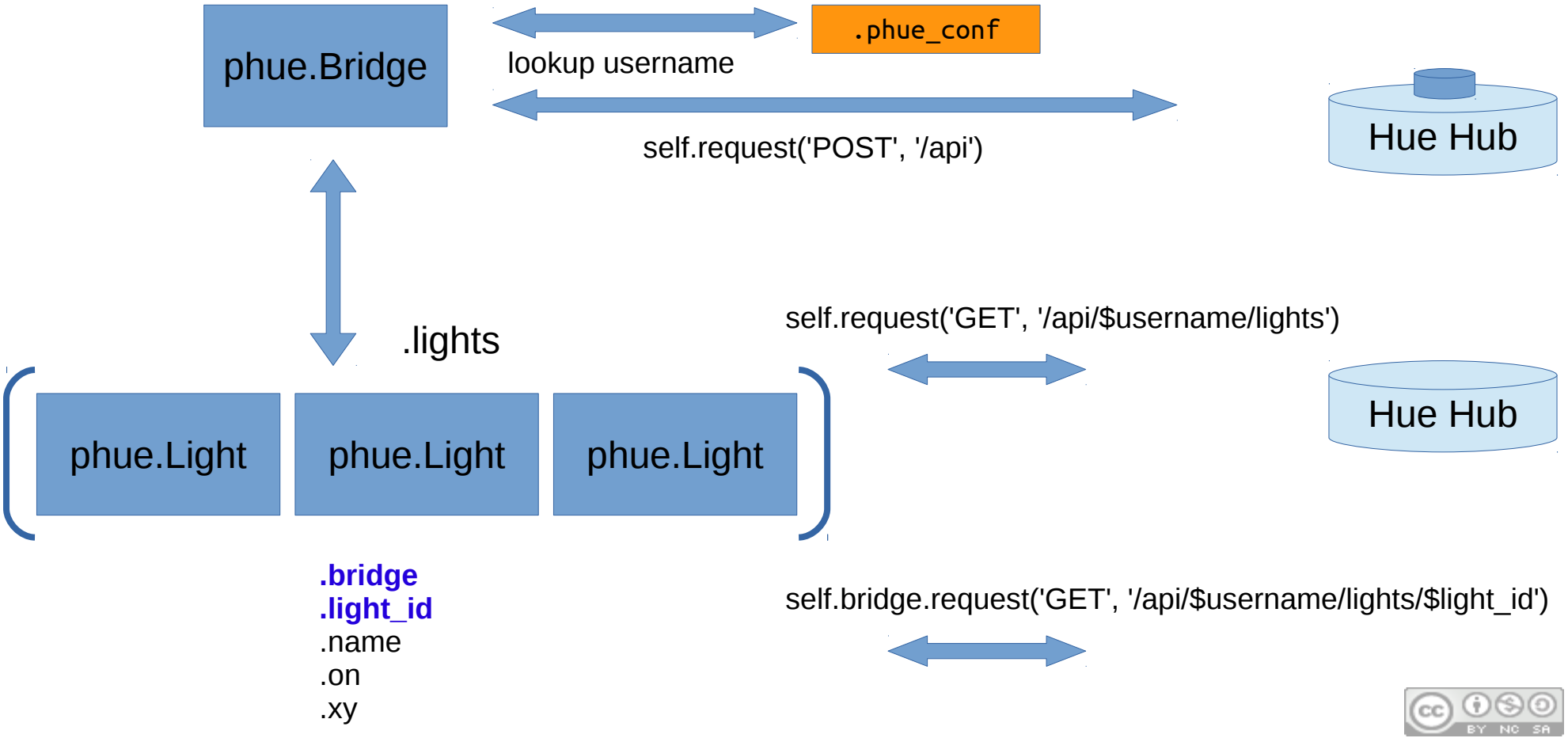


```
    def request(self, mode='GET', address=None, data=None):
        """ Utility function for HTTP GET/PUT requests for the API"""
        connection = httplib.HTTPConnection(self.ip, timeout=10)

        try:
            if mode == 'GET' or mode == 'DELETE':
                connection.request(mode, address)
            if mode == 'PUT' or mode == 'POST':
                connection.request(mode, address, data)

            logger.debug("{0} {1} {2}".format(mode, address, str(data)))

        except socket.timeout:
            error = "{} Request to {}{} timed out.".format(mode, self.ip, address)

            logger.exception(error)
            raise PhueRequestTimeout(None, error)

        result = connection.getresponse()
        connection.close()
        if PY3K:
            return json.loads(str(result.read(), encoding='utf-8'))
        else:
            result_str = result.read()
            logger.debug(result_str)
            return json.loads(result_str)
```

```
class TestRequest(testtools.TestCase):

    def setUp(self):
        super(TestRequest, self).setUp()
        self.home = fixtures.TempHomeDir()
        self.useFixture(self.home)

    def test_register(self):
        """test that registration happens automatically during setup."""
        confname = os.path.join(self.home.path, '.python_hue')
        with mock.patch("phue.Bridge.request") as req:
            req.return_value = [{'success': {'username': 'fooo'}}]
            bridge = phue.Bridge(ip="10.0.0.0")
            self.assertEqual(bridge.config_file_path, confname)

        # check contents of file
        with open(confname) as f:
            contents = f.read()
            self.assertEqual(contents, '{"10.0.0.0": {"username": "fooo"}}')

        # make sure we can open under a different file
        bridge2 = phue.Bridge(ip="10.0.0.0")
        self.assertEqual(bridge2.username, "fooo")

        # and that we can even open without an ip address
        bridge3 = phue.Bridge()
        self.assertEqual(bridge3.username, "fooo")
        self.assertEqual(bridge3.ip, "10.0.0.0")
```

All request calls return same results

What's this software actually look like?



```
    def request(self, mode='GET', address=None, data=None):
        """ Utility function for HTTP GET/PUT requests for the API"""
        connection = httplib.HTTPConnection(self.ip, timeout=10)

        try:
            if mode == 'GET' or mode == 'DELETE':
                connection.request(mode, address)
            if mode == 'PUT' or mode == 'POST':
                connection.request(mode, address, data)

            logger.debug("{0} {1} {2}".format(mode, address, str(data)))

        except socket.timeout:
            error = "{} Request to {}{} timed out.".format(mode, self.ip, address)

            logger.exception(error)
            raise PhueRequestTimeout(None, error)

        result = connection.getresponse()
        connection.close()
        if PY3K:
            return json.loads(str(result.read(), encoding='utf-8'))
        else:
            result_str = result.read()
            logger.debug(result_str)
            return json.loads(result_str)
```

```
LIGHTS1 = {
    u'1': {u'manufacturername': u'Philips',
           u'modelid': u'LCT001',
           u'name': u'Living Room Bulb',
           u'state': {u'alert': u'none',
                      u'bri': 254,
                      u'colormode': u'xy',
                      u'ct': 382,
                      u'effect': u'none',
                      u'hue': 14665,
                      u'on': True,
                      u'reachable': True,
                      u'sat': 156,
                      u'xy': [0.4677, 0.4121]},
           u'swversion': u'5.23.1.13452',
           u'type': u'Extended color light',
           u'uniqueid': u'00:17:88:01:00:d1:fd:53-0b'},
…
}

RESP = dict(GET=dict(), POST=dict(), PUT=dict(),
DELETE=dict())
RESP['GET']['/api/username/lights/'] = LIGHTS1
for key, value in LIGHTS1.items():
    RESP['GET']['/api/username/lights/%s' % key] = \
        LIGHTS1[key]
```

```
class FakeHTTP(object):

    def __init__(self, *args, **kwargs):
        super(FakeHTTP, self).__init__()
        self.call = None

    def request(self, mode, addr, data=None):
        self.call = Request(mode, addr, data)

    def getresponse(self):
        data = samples.RESP[self.call.mode][self.call.addr]
        return StringIO(dump(data))

    def close(self):
        pass
```

```
class TestLights(testtools.TestCase):

    def setUp(self):
        super(TestLights, self).setUp()
        self.useFixture(
            fixtures.MonkeyPatch(httplib, fakes.FakeHTTP))
        self.bridge = phue.Bridge(
            ip="10.0.0.0", username="username")

    def test_get_lights(self):
        lights = self.bridge.get_light_objects('id')
        self.assertEqual(lights[1].name, "Living Room Bulb")
```

Keeping track of progress

Coverage for **phue.py** : 34%

673 statements | 228 run | 445 missing | 0 excluded


```
#!/usr/bin/python
# -*- coding: utf-8 -*-

'''
phue by Nathanaël Lécaudé - A Philips Hue Python library
Contributions by Marshall Perrin, Justin Lintz
https://github.com/studioimaginaire/phue
Original protocol hacking by rsmck : http://rsmck.co.uk/hue

Published under the MIT license - See LICENSE file for more details.

"Hue Personal Wireless Lighting" is a trademark owned by Koninklijke Philips Electronics N.V., see www.meethue.com for more information.
I am in no way affiliated with the Philips organization.

'''

import json
import logging
import os
import platform
import sys
import socket
if sys.version_info[0] > 2:
    PY3K = True
else:
    PY3K = False

if PY3K:
    import http.client as httplib
else:
    import httplib

logger = logging.getLogger('phue')


if platform.system() == 'Windows':
    USER_HOME = 'USERPROFILE'
else:
    USER_HOME = 'HOME'

__version__ = '0.9'

```

Continuous Integration

- Things that happen automatically will happen more often
- If software is easy to test, it's easy to ship
- Many CI tools out there
  - Jenkins
  - Zuul
  - Circle CI
  - etc...

Travis CI Blog Status Help Sean Dague

Search all repositories

My Repositories +

- ✗ home-assistant/home-assistant #14016
  - Duration: 18 min 23 sec
  - Finished: 13 minutes ago
- ✓ sdague/arwn #44
  - Duration: 1 min 57 sec
  - Finished: a day ago
- ✓ sdague/rxv #2
  - Duration: 1 min 38 sec
  - Finished: 3 days ago
- ✗ sdague/home-assistant #67
  - Duration: 14 min 57 sec
  - Finished: 3 days ago
- ✓ sdague/phue #6
  - Duration: 1 min 46 sec
  - Finished: 5 days ago

sdague / phue

build unknown

Current Branches Build History Pull Requests More options

✓ **tests** build framework for testing more complex API interactions

```
This builds a framework for having a set of canned test responses
based on HTTP Requests. It does this by monkey patching out
HTTPConnection, and redirecting it into a static multi level
```

Commit 7026ad9

Compare d8e81a6..7026ad9

Sean Dague authored and committed

#6 passed

Elapsed time 40 sec

Total time 1 min 46 sec

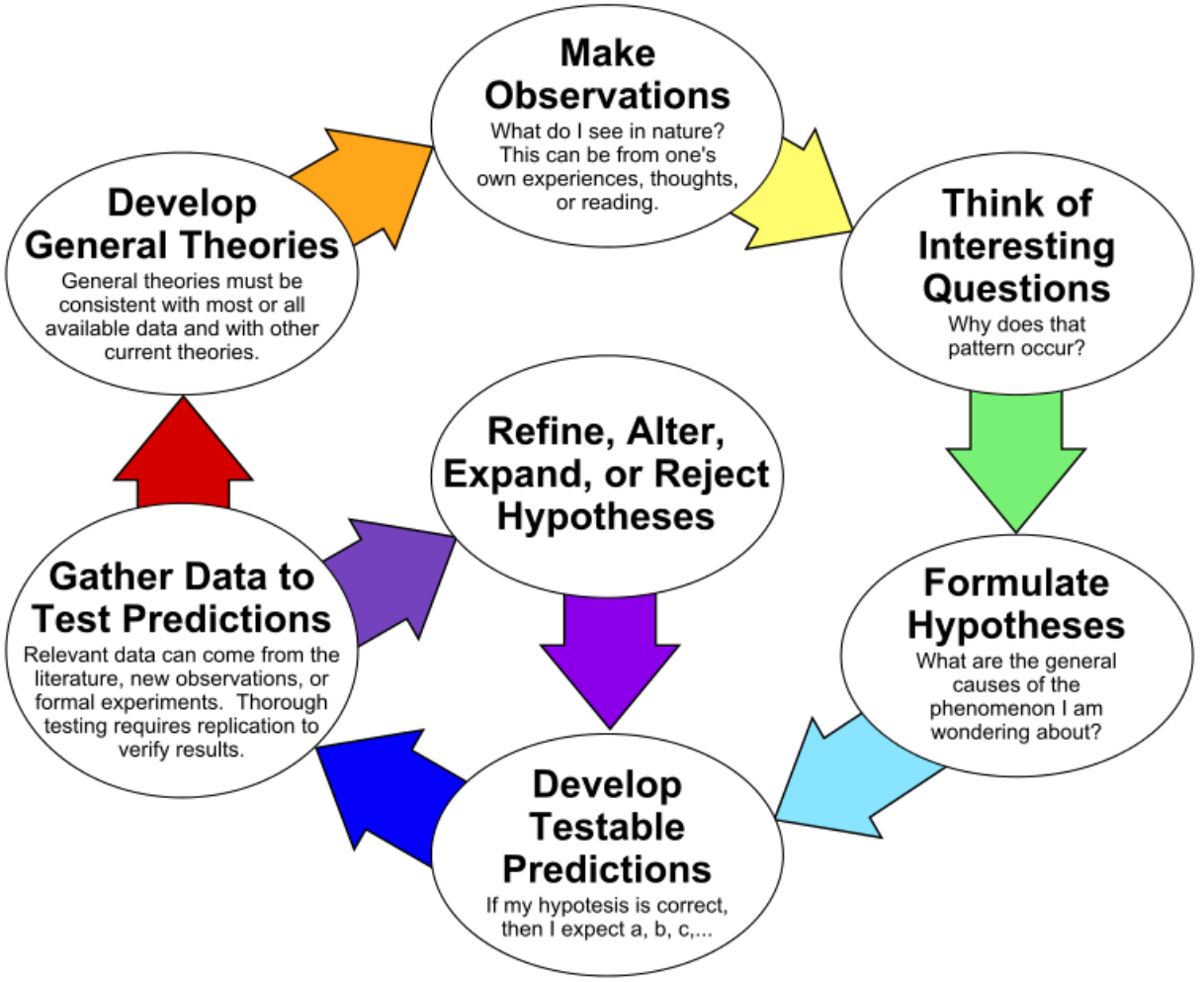
5 days ago

Restart build

Build Jobs

✓	#6.1	Python: 2.7	TOXENV=py27	36 sec
✓	#6.2	Python: 3.5	TOXENV=py35	39 sec
✓	#6.3	Python: 2.7	TOXENV=pep8	31 sec

The Scientific Method as an Ongoing Process



Questions?